Mixed-Projection Treemaps: A Novel Approach Mixing 2D and 2.5D Treemaps

Daniel Limberger, Willy Scheibel, Matthias Trapp, and Jürgen Döllner
Hasso Plattner Institute—Faculty of Digital Engineering
University of Potsdam, Germany
{*daniel.limberger, willy.scheibel, matthias.trapp, juergen.doellner*}*@hpi.de*

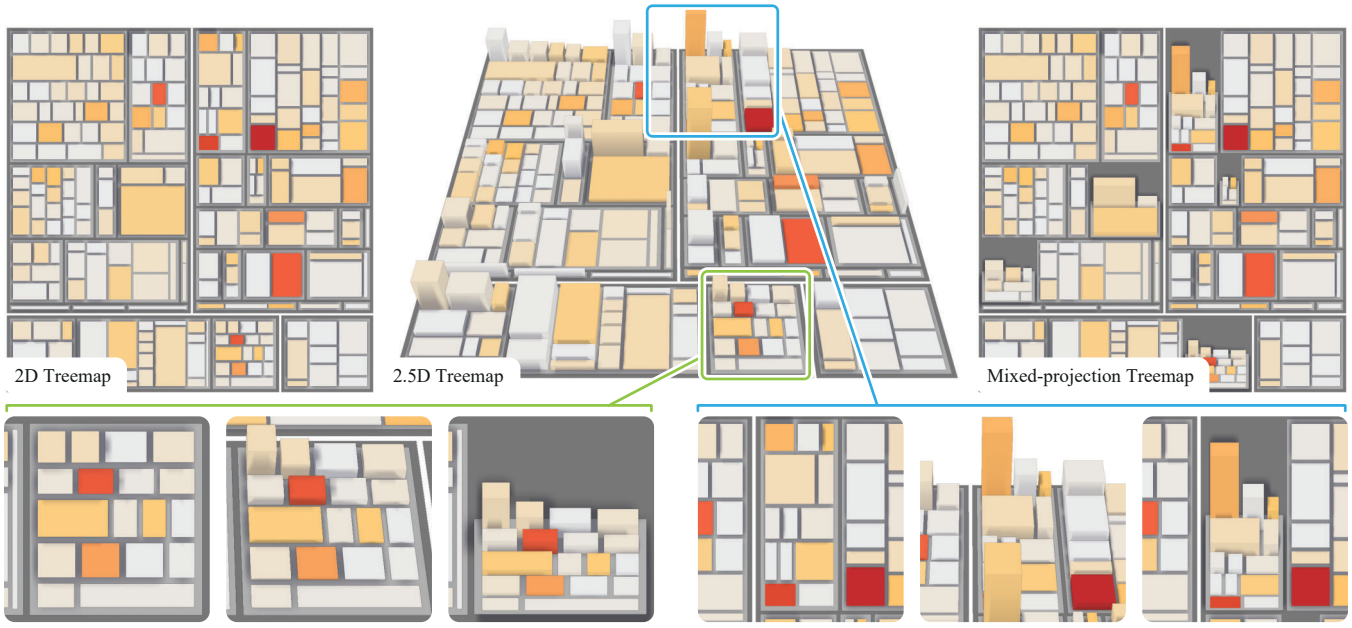

Figure 1. 2D treemap (left) and 2.5D treemap (center) depicting the same data using color and additional height mapping respectively. A mixed-projection treemap (right), in contrast, reduces visual clutter and occlusion. At the same time it exposes important correlations not accounted for in the 2D treemap.

***Abstract*—This paper presents a novel technique for combining 2D and 2.5D treemaps using multi-perspective views to leverage the advantages of both treemap types. It enables a new form of overview+detail visualization for tree-structured data and contributes new concepts for real-time rendering of and interaction with treemaps. The technique operates by tilting the graphical elements representing inner nodes using affine transformations and animated state transitions. We explain how to mix orthogonal and perspective projections within a single treemap. Finally, we show application examples that benefit from the reduced interaction overhead.**

***Keywords*-Information Visualization; Overview+Detail; Treemaps; 2.5D Treemaps; Multi-perspective Views;**

I. Introduction

Treemaps represent a space-filling visualization technique that facilitates display and exploration of non-spatial, attributed, tree-structured data. They map attributes to graphical elements, e.g., rectangles, that are recursively stacked within their parents' elements. As one key advantage 2D treemaps serve as scalable visualization technique for tree-structured data [1]. They are applied in various research and industry applications to provide comprehensive and consistent means for communication and visual analytics [2] of non-spatial, massive, temporal and structural evolving information such as file systems [3] and software systems [4]. 2.5D treemaps extend this concept by introducing height as additional visual variable. The 3D representation, however, leads to increased rendering effort, occlusion, and demands for advanced navigation techniques that counterbalance the advantages of 2.5D treemaps to a certain degree.

The expressiveness of 2D treemaps is limited compared to 2.5D treemaps taking advantage of the third dimension [5]. Apart from the height per graphical element they also provide additional visual variables such as transparency, texture, shading, extended shape, as well as silhouette enhancement techniques [6]. Similar to 2D treemaps, 2.5D treemaps can be efficiently rendered [7] and offer an additional dimension for information display.

2.5D treemaps are limited by a number of drawbacks [5], [8]; such as occlusion, underutilized screen-space, and perspective-foreshortening–problems well known in the field of thematic 3D visualization [9], [10]. Especially occlusion appears to have a "detrimental impact on discovering, accessing, and spatially relating embedded data" [11]. In most applications though, occlusion is usually counterbalanced by a well-tuned *map theme* (a task-centric specification of the attribute mapping to visual variables) and usually does not lead to a "significant loss of information": similar to the use of color, a meaningful use of height facilitates identification of relevant data by means of outlier perception, data range and distribution adjustments, as well as aggregation of irrelevant areas [12]. When applying these established operations the resulting 2.5D treemaps consist mostly of flat cuboids. However, when the input data is partially unknown or the task focuses on dynamic exploration, tuning the overall map theme might not be beneficial or appropriate at all and the visual complexity, e.g., by means of occlusion, can be hindering in the information gathering process.

This paper presents a novel visualization approach for treemaps that combines orthogonal and perspective projections within a single 2.5D treemap. Thereby it strengthens the applicability of treemaps by optimizing their compliance to the visual information seeking mantra "overview first, zoom and filter, then details-on-demand" [13]. For it, rendering and interaction techniques that enable 2D treemaps to get locally extended to 2.5D treemaps (*mixed-projection treemaps*, Fig. 1) are introduced and the following contributions are made:

1) A rendering technique that dynamically mixes different projections supporting seamless integration of 2.5D treemaps for region-of-interests into 2D treemaps to facilitate communication of additional information using more visual variables.
2) An interaction concept that enables users to control the projection by means of manual and automated tilting, either providing explicit control or using animated state transitions, respectively.

We argue that interactive mixed-projection treemaps reduce the need for complex navigation metaphors typically associated with 2.5D treemaps and, further, feature traits common for *focus+context* or *overview+detail* metaphors, i.e., (1) the reduction of visual complexity for overview and context display and (2) the visualization of additional data in focus or detail areas. We demonstrate and discuss these aspects by means of application examples.

II. Related Work

Treemaps are a common technique for space-restricted visualization of tree-structured data. The original algorithm presented by Shneiderman recursively splits parent nodes alternating in horizontal and vertical direction by their child nodes [3], resulting in a 2D layout. By using specific shading [14], margins or insets [15], the treemap can depict topology as well. Often used visual variables include size and color [1] as well as texture [16].

Treemap Layouts: Besides treemap presentations, a basic layout algorithm has to be chosen according to the particular use cases. For example, the choice of the layout algorithm influences the creation and preservation of a mental map for the user of the visualization [17]. One major disadvantage of the Slice'n Dice algorithm is the creation of unfavorable aspect ratios of the node representations. Bruls et al. as well as Bederson and Shneiderman counterbalance this problem using Squarified [18] and Strip Treemap [19] layouts. Latest research focus on the use of space-filling curves and further optimizes the trade-off between readability and stability [20], [21]. Since most of the treemap layout algorithms are designed for 2D presentation, they do not take possible occlusion within 2.5D treemaps into account.

2.5D Treemaps: The first extension of 2D treemap algorithms to the third dimension are presented in the StepTree visualization system by Bladh et al. [22]. It extends the initial approach by stacking the graphical representations of subdirectories. In a comparative study between the StepTree and its corresponding 2D treemap users significantly performed better in tasks of interpreting the hierarchical structure while preserving performance in other interpretive and navigational tasks. In addition to this approach there exist a variety of modifications and extensions, e.g., 3D Polar Treemaps, Treecubes, or Collector's Box Treemaps [5]. In contrast to general 3D treemaps, most of current 3D tree visualization techniques combine a 2D layout and extrude the graphical elements forming the third dimension. Due to the restricted use of the third dimension, we refer to the term 2.5D treemaps. An example on how to adaptively extend the nodes' shape in 2D treemaps by the third dimension was provided by Chaudhuri and Shen [23]. Wettel and Lanza use 2.5D treemaps to create a metaphorical view on software system artifact hierarchies with CodeCities [24] by mapping additional information to the height of leaf nodes. Bohnet and Döllner mentioned the benefits of using realistic rendering effects (e.g., shadows) in 2.5D treemaps to support the users' in distinguishing individual elements [25].

The increasing number of items in large hierarchies leads to perceptional questions about treemap visualizations, e.g., how much effect does the width of a nodes' border have on the ability to extract the hierarchical structure of the underlying data [26]. With an increasing number of items, the need for focus+context as well as overview+detail techniques emerges [27]. Even though the visualization of a wide range of large hierarchical datasets (e.g., software systems, economy data) is frequently studied, the interaction techniques used in this field remains straightforward. Apparent interaction techniques in 2.5D treemaps are the selection of inner nodes and leafs as well as the coloring/highlighting of nodes [24]. Sud et al. present interac-

tive rendering, navigation, and animation of dynamically changing 2D Voronoi Treemaps by using a GPGPU-based rendering technique [28].

Mixed-perspective Visualization: The choice for the type of projection for virtual 3D environments is often either an orthographic or perspective projection [29]. However, depending on the composition and the characteristics of the environment, mixed projections can provide effective communication of the content. For virtual terrain and city model visualization, multi-perspective views with bended projections provides a near-field depiction of high detail and a context area of less detail [11], [30]. Prior to hardware-accelerated rendering techniques for mixed projections were developed [10], these approaches were already applied by artists such as H. C. Berann [31].

III. Visualization

This section (1) introduces a two part tilt operator that uses a sequence of parameterized affine transformations, (2) describes a rendering that relies on a single geometry encoding and uses the exact same pipeline for both 2D and 2.5D nodes, and (3) specifies two interaction modes featuring manual and automated tilt control.

Node-local Tilt Operator

Given a 2D treemap, a tilt denotes the rotation of an inner node (and its children) that enables exploration of the height mapping using either an orthographic or perspective projection. Any tilt comprises the following two node-local operations: a tilt transformation Λ and a tilt projection Γ.

The tilt transformation shifts the node's rotation axis by T_R using a relative offset $\tau \in [-1, +1]$, with -1 shifting to the node's bottom edge and $+1$ to its top edge. It then rotates the node by the tilt angle α using R, and anchors the node by T_A using to a preferred relative location $\upsilon \in [-1, +1]$ (again, -1 at bottom and $+1$ at top, 0 at center). The complete, affine transformation (Fig. 3) is defined as

$$\Lambda = [T_C]\, T_A T_R^{-1} R T_R. \tag{1}$$

T_C thereby denotes an optional translation that reduces occlusion introduced when using a perspective projection. It can be easily derived using the virtual cameras eye position and, e.g., the nodes vertical extend. Also, we advise against deviating from $\tau = 0$ (using perspective projection) since it either introduces more occlusion or reduces the nodes size on display with $\tau > 0$ and $\tau < 0$ respectively.

The tilt projection basically mixes any two given projections P_0 and P_1 with respect to the node local tilt angle and a global angular threshold β and is defined as

$$\Gamma = (1 - t)\, P_0 + t P_1, \tag{2}$$

with $t = \alpha\beta^{-1}$, clamped to $[0, 1]$. If P_1 is a perspective projection P_0 should be the respective orthographic projection, i.e., it should cover the exact same treemap region.

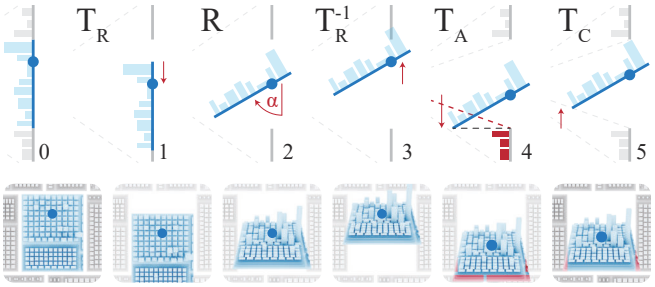

Figure 3. This illustration describes a tilt transformation Λ of an inner-node n with perspective projection applied: n is (1) translated by T_R for (2) rotation by R with $\alpha = 60°$ at the local tilt offset $\tau = 0.5$. After (3) translating n (T_R^{-1}) it is (4) moved to the bottom anchor by T_A with $\upsilon = -1$ and, finally, (5) shifted up again by T_C in order to reduce the occlusion caused by the perspective projection. Step (5) is only required when using orthogonal projection.

Even though, both transformation and projection are applied per-node and can be at different states at any time for any node, they are restricted to global parameters for β, τ, and υ provided by the map theme. For a seamless tilt transition only α has to be increased, starting at $\alpha = 0$ for no transition up to the desired, final tilt angle.

Rendering

For the interactive use of mixed-projection treemaps a real-time rendering based on attributed vertex clouds [32] is used. It allows for efficient GPU encoding of hierarchical information and facilitates interactive, dynamic attribute mapping, and on-the-fly, attribute-based geometry instantiation. The use of the node-local tilt operator allows us to rely solely on a 2.5D GPU encoding for simultaneously rendering some nodes in 2D and others in 2.5D (either orthographic or perspective). This means that the height mapping is always enabled, but not noticeable on nodes that are not titled. Furthermore, the same rendering pipeline can be applied to all nodes. It ultimately enables a seamless transition between 2D and 2.5D as well as orthographic and perspective projections, especially since the results of screen space effects remain consistent w.r.t. shading, local ambient occlusion, shadows, contouring etc.

Interaction

Navigation in 2.5D treemap can be implemented by means of a landscape-metaphor or world-in-hand metaphor. In the case of mixed-projection treemaps, however, a simple zoom and pan metaphor is sufficient for navigating the treemap. Based on the parameterization introduced previously, a user can interact directly with treemap nodes based on direct manipulation metaphors. For it we suggest two similar tilt modes: *manual tilt* and *automated tilt*. A manual tilt enables the user to increase and decrease the tilt angle of any node seamlessly (e.g., via click/touch to an inner node and vertical drag down). An automated tilt enables the user to invoke a preset tilt angle or un-tilt any node by a single input event

(e.g., click/touch to an inner node). To avoid abrupt tilt angle changes, an animated transitions with arbitrary easing can be applied. Due to the simplicity of both tilt interactions, additional interactions (e.g., filtering) can be integrated more easily. Both modes can also be combined using applying some type of mode switch. Keep in mind though that the tilt is not designed for nested application.

IV. Results

When using mixed-projection treemaps to visualize tree-structured data, the designated use case follows a multi-step process. First, the user gains overview using the 2D top view on the treemap rendered with orthographic projection. For all areas-of-interest, the user can select the inner nodes. Those selected nodes are then tilted into a 2.5D presentation with additional visual variables. The prototypical implementation is an extension to a 2.5D treemap visualization system by means of (1) the management of selected inner nodes for the detail view, (2) computing the nodes' geometry tilting, (3) efficient image synthesis using an attributed vertex cloud of axis-aligned cuboids where (4) the tilt is applied during geometry instantiation [32].

Application Examples

Mixed-projection treemaps support in getting an overview over a dataset first and finding relevant nodes afterwards by enabling the third dimension on-demand. As examples of real-world tree-structured datasets we chose a population count dataset and a software system dataset.

Population Count Dataset: The first dataset is the population count of the zoo in Munich from 2014 (Fig. 1). The dataset contains 247 nodes whereby each node represents one species. The footprint size of a visual item refers to the population count of a species, the color refers to the extinction threat index, and the number of born animals in the last year is mapped to the height. This map theme facilitates the detection of recovering species.

Software System Dataset: The second dataset represents an analysis of the open-source project POCO[1] (Fig. 6). The dataset contains 5775 nodes. Each node represents one source code file. The number of real-lines-of-code is mapped to the footprint size, the color refers to cyclomatic complexity, and the average nesting level of a source code file is mapped to the height. This map theme is primarily used to detect source code files with disproportional nesting level in comparison to the implemented business logic.

Discussion

While the process is set, the actual depiction of selected nodes is subject to discussion. There are different approaches for setting the tilt anchor, angle, and additional occlusion-reducing rotation. To support users in maintaining their mental map, animated transition and different types of perspectives can be applied.

Perspective vs. Orthographic Projection: Due to perspective foreshortening in a perspective view (the default for virtual 3D environments), objects that are far form the virtual camera away are rendered smaller than closer ones. Alternatively, in an orthographic view, all objects appear at same scale. Since the user preference of projection may differ, both options are supported: (1) perspective projections provide additional information about depth and are often easier to interpret, and (2) orthographic projections facilitate comparison of nodes w.r.t. height and area. Further, an orthographic projection renders no occlusion to neighboring nodes in the overview.

Tilt Anchor: After tilting the inner nodes, the inclination introduces unused screen space. This may be partially covered by higher nodes but the overall impression remains and even supports the identification of these nodes as tilted (and thus, conveying height information). To exploit this space further, the tilted node can be shifted within the original bounding rectangle (Fig. 4).

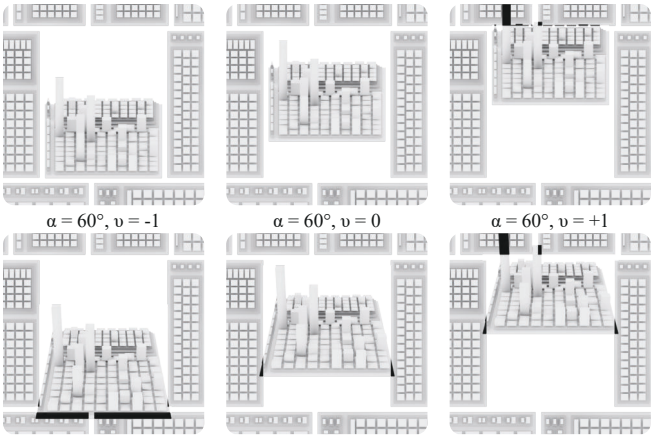

Figure 4. Orthogonal (top row) and perspective (bottom row) projections anchored at different locations, all tilted by 60 (overdraw colored black).

During experiments we made the following observation: the lower alignment seems to be most beneficial as the risk of occluding nearby nodes is minimized (compare to the right pictogram with top alignment; overdraw is colored black) and there is still sufficient space for possible additional information display (as opposed to the center alignment with justified whitespace, compare center pictogram).

Tilt Angle: The effective tilt angle is subject to inner-node complexity and height-value distribution. Although angles between $0°$ and $90°$ are possible, a tilt degree of $\alpha = 0°$ results in no additional height information and a tilt of $\alpha = 90°$yields high occlusion (Fig. 5). We found tilt angles between $30°$ and $60°$ to be reasonable supportive. We also found that a slightly different shading of the cuboids' lateral surfaces servers as additional tilt cue.

[1]GitHub project available at github.com/pocoproject/poco; Analysis of revision from March 24 in 2009.

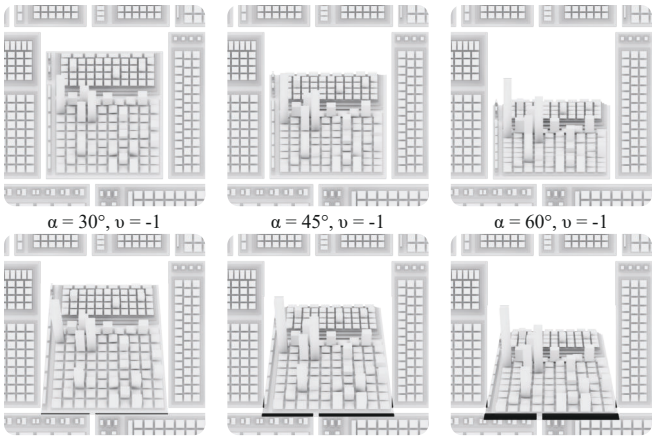

Figure 5. Orthogonal (top row) and perspective (bottom row) projections at different tilt angles, all anchored at the bottom (overdraw colored black).

Tilt Transitions: Even though the mixed-projection treemap is designed for static display, the use of manual or animated transitions can help a user to maintain its mental map during the state change from overview to detailed depiction of an inner node and is highly suggested.

V. Conclusion and Future Work

This paper presents how to combine 2D and 2.5D treemaps. The concept is based on in-situ mixed projections, i.e., the dynamic adjustment of and interpolation between orthographic and perspective projections of components of a single treemap based on regions of interest, which can be interactively selected by a user. The combination strengthens the particular advantages of each projection technique: the overview and simple interaction with tree-structured data using 2D treemaps and the advanced capabilities of presenting additional data attributes using 2.5D treemaps. To some extent it counterbalances the limitations inherent to 2.5D treemaps, e.g., occlusion and perspective-foreshortening. It also simplifies the user interactions required for 2.5D treemaps. Preliminary results obtained by using our prototypical implementation indicate a simplified treemap exploration and effective comparison of additional data mapped to height. For future work we plan to take into account the available screen space for display of additional information using data charts such as histograms.

Acknowledgment

This work was funded by the Federal Ministry of Education and Research (BMBF), Germany, within the InnoProfile Transfer research group "4DnD-Vis" (www.4dndvis.de).

References

[1] J.-D. Fekete and C. Plaisant, "Interactive information visualization of a million items," in *Proc. IEEE INFOVIS*, 2002, pp. 117–124.

[2] A. M. MacEachren, *How Maps Work: representation, visualization, and design*. Guilford Press, 1995.

[3] B. Shneiderman, "Tree visualization with treemaps: A 2d space-filling approach," *ACM ToG*, pp. 92–99, 1992.

[4] D. Limberger, B. Wasty, J. Trümper, and J. Döllner, "Interactive software maps for web-based source code analysis," in *Proc. ACM Web3D*, 2013, pp. 91–98.

[5] H.-J. Schulz, S. Hadlak, and H. Schumann, "The design space of implicit hierarchy visualization: A survey," *IEEE TVCG*, vol. 17, no. 4, pp. 393–411, 2011.

[6] D. Limberger, C. Fiedler, S. Hahn, M. Trapp, and J. Döllner, "Evaluation of sketchiness as a visual variable for 2.5d treemaps," in *Proc. IV*, 2016, pp. 183–189.

[7] M. Trapp, S. Schmechel, and J. Döllner, "Interactive rendering of complex 3d-treemaps with a comparative performance evaluation," in *Proc. GRAPP*, 2013, pp. 165–175.

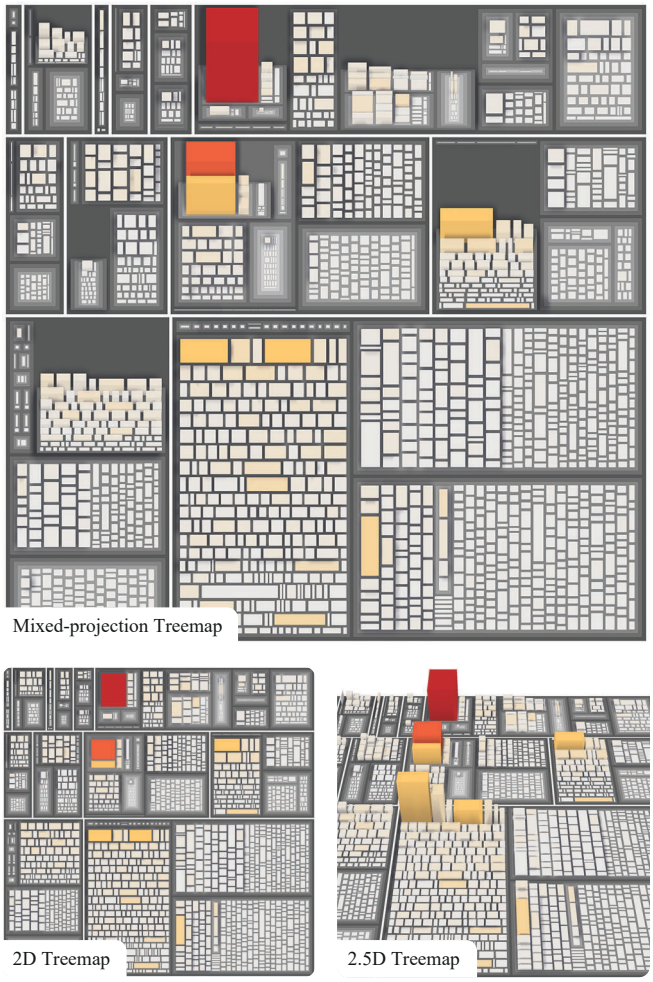

Figure 6. A 2D treemap, 2.5D treemap, and a mixed-projection treemap (with various nodes tilted), all three of which depict the same POCO software system.

[8] F. van Ham and J. J. van Wijk, "Beamtrees: Compact visualization of large hierarchies," *SAGE Information Visualization*, vol. 2, no. 1, pp. 31–39, 2003.

[9] N. Elmqvist and P. Tsigas, "A taxonomy of 3d occlusion management for visualization," *IEEE TVCG*, vol. 14, no. 5, pp. 1095–1109, 2008.

[10] H. Lorenz, M. Trapp, J. Döllner, and M. Jobst, "Interactive multi-perspective views of virtual 3d landscape and city models," in *The European Information Society*. Springer, 2008, pp. 301–321.

[11] M. Röhlig and H. Schumann, "Visibility widgets: Managing occlusion of quantitative data in 3d terrain visualization," in *Proc. ACM VINCI*, 2016, pp. 51–58.

[12] D. Limberger, W. Scheibel, S. Hahn, and J. Döllner, "Reducing visual complexity in software maps using importance-based aggregation of nodes," in *Proc. IVAPP*, 2017, pp. 176–185.

[13] B. Shneiderman, "The eyes have it: A task by data type taxonomy for information visualizations," in *Proc. IEEE Visual Languages*, 1996, pp. 336–343.

[14] J. J. van Wijk and H. van de Wetering, "Cushion treemaps: Visualization of hierarchical information," in *Proc. IEEE INFOVIS*, 1999, pp. 73–78.

[15] H. Lü and J. Fogarty, "Cascaded treemaps: examining the visibility and stability of structure in treemaps," in *Proc. GI*. Canadian Information Processing Society, 2008, pp. 259–266.

[16] D. Holten, R. Vliegen, and J. J. van Wijk, "Visual realism for the visualization of software metrics," in *Proc. IEEE VISSOFT*, 2005, pp. 1–6.

[17] K. Misue, P. Eades, W. Lai, and K. Sugiyama, "Layout adjustment and the mental map," *Elsevier JVLC*, pp. 183–210, 1995.

[18] M. Bruls, K. Huizing, and J. J. van Wijk, "Squarified treemaps," in *Proc. Joint EG / IEEE TCVG VIS*, 1999, pp. 33–42.

[19] B. B. Bederson, B. Shneiderman, and M. Wattenberg, "Ordered and quantum treemaps: Making effective use of 2D space to display hierarchies," *ACM ToG*, vol. 21, no. 4, pp. 833–854, 2002.

[20] Y. Tu and H.-W. Shen, "Visualizing changes of hierarchical data using treemaps," *IEEE TVCG*, vol. 13, no. 6, pp. 1286–1293, 2007.

[21] S. Tak and A. Cockburn, "Enhanced spatial stability with hilbert and moore treemaps," *IEEE TVCG*, vol. 19, no. 1, pp. 141–148, 2013.

[22] T. Bladh, D. A. Carr, and J. Scholl, "Extending tree-maps to three dimensions: A comparative study," in *Proc. APCHI*. Springer, 2004, pp. 50–59.

[23] A. Chaudhuri and H.-W. Shen, "A self-adaptive treemap-based technique for visualizing hierarchical data in 3d," in *Proc. IEEE PacificVis*, 2009, pp. 105–112.

[24] R. Wettel and M. Lanza, "Visualizing software systems as cities," in *Proc. IEEE VISSOFT*, 2007, pp. 92–99.

[25] J. Bohnet and J. Döllner, "Monitoring code quality and development activity by software maps," in *Proc. ACM MTD*, 2011, pp. 9–16.

[26] N. Kong, J. Heer, and M. Agrawala, "Perceptual guidelines for creating rectangular treemaps," *IEEE TVCG*, pp. 990–998, 2010.

[27] H. Hauser, "Generalizing focus+ context visualization," in *Scientific visualization: The visual extraction of knowledge from data*. Springer, 2006, pp. 305–327.

[28] A. Sud, D. Fisher, and H.-P. Lee, "Fast dynamic voronoi treemaps," in *Proc. IEEE ISVD*, 2010, pp. 85–94.

[29] H. Piringer, R. Kosara, and H. Hauser, "Interactive focus+context visualization with linked 2d/3d scatterplots," in *Proc. IEEE CMV*, 2004, pp. 49–60.

[30] S. Pasewaldt, M. Trapp, and J. Döllner, "Multiscale visualization of 3d geovirtual environments using view-dependent multi-perspective views," *Journal of WSCG*, vol. 19, no. 3, pp. 111–118, 2011.

[31] H. C. Berann, "The world of H.C. Berann," Online. http://www.berann.com/. [Online]. Available: http://www.berann.com/

[32] W. Scheibel, S. Buschmann, M. Trapp, and J. Döllner, "Attributed vertex clouds," *GPU Zen (to be published)*, 2017.